スペイン語テキスト処理の実際：
単語検索の諸問題

出　口　厚　実

Dos programas para la búsqueda de palabra españolas en los ficheros de texto del MS—DOS

Atsumi DEGUCHI

A pesar de un diluvio de programas de aplicación de que gozamos ahora, no hay herramientas o métodos cómodos de localizar la aparición de palabras dentro de ficheros de texto en español, al menos para el sistema operativo MS—DOS acualmente disponible en los ordenadores de la serie NEC 9800.

En este artículo se presentan los listados de los dos pequeños programas llamados WSCH.SNO and WFIND.COM que tienen la función de localizar una palabra específica en los ficheros de texto del MS—DOS formateados en STX, y dar como salida (en pantalla o redirigible hacia cualquier fichero) para cada aparición de la palabra buscada, la oración completa que la contiene, precedida por sus números de página y línea correspondientes.

El primer programa está escrito en el lenguaje SNOBOL4 (véase listado (7)), y el otro está codificado para ser ensamblado con el Turbo Assembler ver 2.0 (véanse listados (8a) y (8b)) que permite producir el programa ejecutable en formato .COM.

0．は　じ　め　に

パーソナルコンピュータを利用すれば、従来の手作業では考えられなかったような大量の文章資料を参照・調査したり、様々なデータ処理が効率よく簡単に行えるようになっていると信じる人々は少なくない。コンピュータが既定としている英語・日本語に関してはそのような近未来もあながち非現実的とは言えないかも知れない。しかし、文章の編集・整形・印刷といったワープロ機能に限定するならばともかく、何か独自の加工や分析を他言語のテキストに対して試みようとすれば、なお解決しなければならない問題が山積している。そのうちのいくつかについて拙稿（1989, 1990）で触れた。それらは、現在、わが国で最も多く使用されているＮＥＣ　ＰＣ－9800シリーズ及びその互換機のハード仕様や関連するＯＳ規格の日本でのインプリメントの仕方などに起因する制約のために、資料をデータ化する前段階で整備されなければならない、外国語文を受付けさせるための基礎的な環境づくりについてであった。

テキストが日英語でないという理由のために要求されるソフトウエア上のハンディを背負わねばならないのとは別に、何語であれ、およそ自然言語テキストをごく当たり前に扱いたいという欲求に対しても、元々障壁が存在する。コンピュータは、元来、厳密に定格化された数値や文字列、それらの構造化された組合せをデータとして処理するのが主目的であったので、その方面にはきめ細かな配慮もなされ、既製品ソフトの数も多い。だが、文書作成から一歩踏み出して、文章体資料そのものを言語データとして、検索・集計・収集などの分析の対象としようとするときには、意外に貧弱な基盤と道具立てしか用意されてないようである。

自然言語のテキストを文法事項の調査や文体研究に役立てることが、ますます底面積を広げつつあるパソコン利用の現場にあっても、相変わらず少数派であるのは明かである。外国語学・文学の教育、研究に携わる人々の数を想像すれば、しかし、決して特殊なパソコン活用法とは言えないはずである。現に、筆者の周辺においても何らかの形でパソコンを仕事に使うという同僚が急増しているのが実状である。外国文データにどの様な情報を捜し求め、何に活用するかの、２次、３次的な利用目的には千差があり、個人により多岐に拡散するだろう。しかし、多くの人々に共通な種類の基本作業も少なくないはずである。各人が必要な処理毎にそのためのソフトを新作しなければならないとすれば、いたずらに時間と労力を費やし、本来の動機であるデータ処理の「省力化」や「時間削減」はすぐさま帳消しになってしまう。その際、行いたい仕事の１部分でも既存のソフト・ツールを組み合わせて利用できれば、作業は大幅に軽減される。筆者自身、テキスト処理のための小ソフトで、もし手元にあれば使いたいと思うようなものが、数え切れないくらいあるし、ＰＤＳとして流通する特殊なパーツソフトが、偶然、自分の目的にかない重宝したケースが再三ある。こういった簡便な小道具類は用途別に多種多数必要であり、その大部分は専門プログラマを煩わさなくても、われわれ自身の手で作成することが出来る。それどころか、扱う外国語の特殊性の故に、利用者のみが最適の開発者になれる場合すらある。

肝要なことは、同じ様な目的のために似たような用具を異なる人々が各所で自己用に作り専有

するのは不毛な重複ではないかという点である。この種の小プログラムの所在や入手方法についての情報の風通しを良くし、お互いに公開し誰にでも利用できるような体制を築き上げるべきではないか思う。現在のところ、学術研究のためのものであっても、コンピュータ用のデータやプログラムを所有者個人の専用に囲い込んでしまおうとする風潮があることは、残念ながら、否定できない。利用者数の増大につれて、今後少しづつ一般公開利用の原則が理解され、根を下ろして行くとともに、このことがパソコンの利用者を拡大し、各種のツールの開発が促されるという相乗効果を生むことを期待したい。

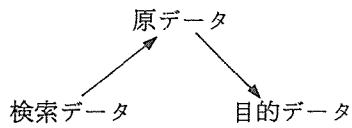
1．語形の検索

文章テキストをコンピュータで扱う場合、最も利用頻度の高いのは「語」または語形の検索ではないだろうか。「検索」は機械化データ処理の要でもあり、エディタ、ワープロ、データベース管理システム（DBMS）などで必須の目玉機能でもある。しかし、実際に外国語文 Corpus から特定の単語を検べ出し、語周辺の情報を集めようとしてみれば、これらの市販ソフトウエアのみではかなり不便であることがわかる。本稿ではその問題点を概括して、様々な空隙を埋めるユーティリティが案出されるべき余地のあることを示唆するとともに、必要に迫られて編み出した筆者自身の解決法の例を2つ紹介する。

2．データのタイプと単位

検索作業におけるデータは少なくとも次の3種類を区別しなければならない。

（1）



原データは検索ソフトが処理の対象にして走査するデータで、その中から検索データと一致する情報を含むある範囲のデータを目的データとして取り出すのが検索である。目的データとは検索データ＋コンテキストと考えることが出来る。本稿ではスペイン語の文章が手書きや印刷された通常文書と出来る限り一致した平文テキストを「原データ」とする場合に絞って考察する。原文テキストに最も忠実な電子化データは、文書全体をグラフィックイメージ化した記録形式であろうが、古典作品の校訂から一貫して扱うケースを除いて、文法や文体の研究では不必要である。平文テキストが具体的にどの様なスタイルをとるかは、機械可読テキストの利用目的に依存する。用途が予め狭く限定されておらず、非構造的で、普通の文章と同様に自然に読み得る形式のものをここでは平文と呼んでおく。テキストの加工度を高めてコンピュータのソフト側が要求する規格に大幅に歩み寄ったフォーマットは、それにより全文が復元可能でも、2次的な非平文テキス

トと見なされるだろう。　我々が外国語テキストを資料とするとき、最も扱いやすいのがこのような平文であり、保存・蓄積の形式として妥当であるだけでなく、閲覧・修正などが容易である。処理系が規定する標準的なテキストファイル上に平文データを作れば、「原データ」としてかなり広範な可搬性を持つという点でも有利である。データが平文であるということは、最小単位が１文字で、その上位の構成単位としての固定長が無い点も融通に富み、便利である。データを作成中にいかなる部分で中断しても（１文字の入力中は中断不可能だから）、「原データ」としての無欠性は損なわれないからである。具体的な１例を示すならば、小規模なテキストの処理には、以下のような書式の平文データで実用上、さほど支障はないようである。

（2）

1．ページング

テキストの各頁冒頭行の第１桁を一定の記号ではじめ、その後に原文ページと同一値を転記する。

2．行の字数

テキストの行字数は原文の１行字数と原則として一致させる。特殊文字が複文字化されているときは、平文テキストで字数が増えるが、１行の情報量は等しく保たれる。ハイフン改行されているときは、そのままハイフンを明示して改行する。

3．行番号

データの中に行番号を含めない。テキストを拾い読みする際に、何頁何行目かが瞬時に読み取れないという難点が生じるが、エディタを行表示モードにして、頁頭から相対計算するなどの解決法がある。行番号をテキスト中に明記するためのコストと記録量の増大というマイナス要因を斟酌した結果でもある。

4．テキスト内見出し

無ページ文書、あるいはページ付けが殆ど無意味な原文については、ページング識別位置に、章、節、段落などのタイトル文字列を記入する。見出し部分を無視したいときは注釈行とする。逆に、通常のテキスト内文章として処理する場合は後続の単位と分断するための何らかのデリミタを付加する必要も生じる。

5．注釈行

行頭の一定の記号は注釈行と解され、その行が処理の際に無視されるものとする。原文書に関する各種の書誌情報やテキスト作成日時、作成者 etc. の備忘記録を記しておくために用いる。なお、大規模なテキスト集合を一度に走査するときは、文書名、著者名などの情報が「目的データ」の位置として定格化される必要があるが、筆者の場合のように、個人の手作業で入力出来る20－30頁の程度の小規模データ処理では、これらを注釈行内に記すだけで十分である。

次に「目的データ」のタイプについて述べる。「原データ」が文章資料体であるから、段落、

文、節、前後一定語数など様々な大きさが取り出し単位となり得る。しかし、利用価値、サイズとしての手頃さから見て、我々にとって最も妥当な単位は「文」であろう。文の定義は通常の正書法に基づいて概ね自動処理が出来る便宜的なもので構わない。韻文に関しては、行と節の両方を使い分ける必要があるかも知れない。いわゆるコンコーダンスで常用されるキー語の前後＊＊字という定字数のコンテクストを単位化する形式を用いることもできる。しかし、字数による機械的な幅指定は必要な文脈が途切れたり、反対に不要なデータが付随するといった弊害を伴うので、文法的考察を中心目的とする場合は「文」が最適な区切りとなる。定幅のコンコーダンスは、むしろ、第1次情報として目星をつけるべき場所を教えるインデックス機能が主であるように思われる。

「検索」の単位は「語」であるという前提で稿を進めてきたが、実際の検索現場でユーザ側の使いやすさから言えば、1つの語形の所在を見つけ出してくれるだけでは不十分と思われる。例えば、規則変化動詞の comer が使われている文をすべて抽出したい、といった検索法は決して贅沢な要求ではないだろう。ところが、語形よりもわずかながら抽象的な「語」レベルの検索が配慮されている既製品ソフトの例は殆ど無いのではないか。かろうじて、特定の名詞・形容詞の用例を捜すときに後方曖昧一致でおよその検出が可能な位である。異形態でなくとも、異なる語（形）の一括走査（論理和）も用意しておきたい機能である。単一キーワードの検索を繰り返せば目的は達せられるが、わずか数語であっても、1度に探索結果が得られ、かつ処理時間が短縮される方が望ましい。「目的データ」（主に「文」）内に複数のキーワードが同時に含まれる条件（論理積）の検索も必要であろう。さらに欲張るならば、複数キー間の距離ゼロの場合や可変数の任意語を介在を認めるなどある程度の条件付き共出関係で限定したいケースも起こり得る。そのほか、「検索データ」に関しては検索者がカスタマイズできる更に複雑な構造や、絞り込みによる連続検索などへの対応が出来れば一層有難い。

3．既存の関連ツール

前項で見たような平文テキストの中から特定の語形を検索する手短かな方法を検討してみよう。まず第1に、ＯＳを購入する際、パッケージに含まれて来るコマンド群の中の、文字列検索命令を用いる方法が考えられる。MS－DOS の場合、FIND という外部コマンドが用意されている。実行ファイルは約8kb とスリムな上に、処理速度は高速で、フィルタとして使用できるのも好都合である。ただ、一番問題なのは「目的データ」がテキストの1行に固定されているという点である。該当キー文字列が含まれるテキストの1行と、その先頭からの相対行位置のみしか出力されない。この制限のために FIND は、自然言語テキスト処理には非常に補助的な用具としてしか役立たないのではないだろうか。もう1つはこのコマンドの「検索データ」は単語ではなく、文字列であり、文字列としての完全一致や部分一致を調べる機能しかない欠点である。例えば、前置詞 con を検索すれば、acontecer, descontento,...など、con を部分文字列として含む語ま

で過剰に発見してしまうのである。逆に、行末がcon–で終わり、次行に残りの綴りtentoが渡っている単語が「原データ」にあると、FINDでcontentoを捜しても見つけ出されず、反対に、このケースがconやtentoという文字列にマッチする不都合が生じる。

このように単語検索では過不足を生じ、不正確な結果しか得られないという事実や、文単位を扱い得ないという制限はパソコン用語で使われる「テキスト」の概念が我々の平文テキストと食い違うためである。コンピュータでのテキストは画面デバイスに表示可能な記号列の集合体といった意味であり、暗黙裡に行単位のデータ構造が意識されている。テキスト編集とは主にプログラミング言語のソースの作成を指していたのである。FINDコマンドも基本としてこのようなプログラムソース編集上のツールの１つであるから、自然言語の、しかも外国文テキストの処理には、元々、不向きなのだと考えられる。

文字列パタンの探索ツールとしてGREP(1)とその改良・修正バージョンがかなり普及している。残念ながらGREPもまたＦＩＮＤの延長線上にあり、やはりプログラマ指向のテキスト処理が想定されている。前方・後方曖昧一致や、１文字の任意マッチなど、位置毎の文字種が特化できる、「正規表現」と呼ばれるパタンの書式を指定することができ、この記法には処理系を超えたかなりの互換性がある。また、FINDと異なり、文字列を単語として検索するためのオプションも備わっている。前述の、単語conと完全一致する語を含む行をサーチするには次のような命令を与えればよい。

（３）grep　－iwn　con　ファイル名

もっとも、「語」のデリミタの既定値はＡ－Ｚ.9－0であるため、conをSTX形式（cf. 出口1989, 1990）のファイルから捜す場合、やはり過剰な誤一致を来すことが有り得る。すなわち、con~acと表記された語に標準的なワードサーチではconという単語がマッチするからである。また、スペイン語テキストでdéがde'と表記されているファイルに対して、（４）の命令を与えると、darの接続法現在のdéを含む行も該当するものとして検出してしまう。

（４）grep　－iwn　de　ファイル名

ＳＴＸ形式のように語中にdiacriticとして'~^が使用されている「原データ」に対しては、従って、オプションを次のように変更しないと正しい結果が出ない。

（５）grep　－w[^a－z9－0'~¥^]n　　検索語　ファイル名

いずれにせよ、GREPもまた２行にまたがる単語を正しく認識することができないため、2.で見たような平文テキストをうまく扱えないという点ではFINDと変わりない。

検索用具として最も簡便に利用できるのはエディタとワープロかも知れない。市販エディタでは一般文字列検索とワードサーチは区別され、どちらも可能である。もっとも、GREPのように正規表現まではサポートされず、単純なワイルドカードのみが使用できるようである。また、「語」の既定定義を変更できないのは不便で、スペイン語表記に何らかの代字を併用した「原データ」には実際、完全な検索を期待できないということである。該当語の含まれる１行だけを示す

FINDやGREPと違って発見位置をカーソルで教えるエディタではその前後のコンテキストを任意なだけブロック化して、別ファイルに切り出す機能が備わっているから、テキストの編集過程などで簡単な検索をして、メモを作成する程度には役立つと思われる。しかし、発見箇所が10ヵ所あるいは数十にも及んだ場合、各部分を抜き出す操作はきわめて煩瑣となり、もっとも避けたい単調な作業である。また、検索を始める前に、1度エディタを起動して対象ファイルを読み込む必要があるのも厄介で、この点ではFINDやGREPのように、直接「原データ」を走査し始め、結果のみを、かつ結果をすべてリダイレクトできるソフトの方が優っている。ワープロにおける検索もエディタとほぼ同様な使いにくさが見られるようである。スペイン語文を編集できるとうたわれている、国内の2種類の有名ワープロソフトで実験してみた範囲では、そのような印象を持った。その1つでは、語検索が不可能で、FINDコマンドと同等な文字列の包含一致検索しかできなかった。1字ワイルドカードとその多重使用がサポートされてはいるが、我々の用途にあまり役立つとは思われない。もう1つの商業ソフトでは、マニュアルにも書かれ、また選択メニューにも明示されている“単語全体との一致”（ワードサーチの意と理解される）が正しく動作せず、このモードで、例えば、deを検索すると、del, conde, candela,...etc.も見つけ出してしまうのには驚いた。

4．部分的な解決をめざして

平文テキストを「原データ」として、その中から「文」を「目的データ」単位として単語検索を行うのに役立つ手段は、少なくとも、日常的なパソコン使用環境に備わっていないことがわかった。広いユーザを想定した複雑高度な性能をもつ検索システムは専門家による開発を待たねばならないだろうが、使用頻度が高く、基本的な機能として要請されるけれども、既製のソフトではサポートされていない次の諸条件は差し迫った解決が必要と思われる。

（6）1．スペイン語の書記体系に合わせた特殊字を含む平文テキストを検索できる。
2．行末で途中改行された単語（ハイフンで区切られているものとする）を認識する。
3．該当単語の含まれる「文」全体を表示する。
4．発見事例の所在を、ページ・行で表示する。

そこで、最小限度、上の4項目を満足させるスペイン語の簡易単語検索プログラムを自作して、間に合わせることにした。（6）1.については出口（1989, 1990）で検討したＳＴＸ形式のファイルのみを対象にする。「原データ」のページ数は、（2）1.の要領で［　］内に書かれているものとする。

以下で紹介するのはSNOBOL4とアセンブリ言語で作成した、ほぼ同じ目的のための2種類のプログラムである。この2つの言語を選んだ理由には次の事情がある。高級言語の中で、特に、文字列処理の便宜が考慮されていて、そのための組込み関数が豊富で、プログラミングが易しいという評判のSNOBOL4の実効性を確かめてみたかったというのが1つである。一方、同じ機

能を出来る限り、高速でかつ小さなプログラムで実現したい必要性も感じており、SNOBOLとは対照的な低水準言語によるコーディングも行い比較したいと思っていた。

SNOBOL4の特徴や応用例に関しては多くの文献があり（Cf. Griswold et al (1970), Butler (1985), Hockey (1985))、改めて解説する必要はないだろう。今回、使用したのはMS−DOS版(2)であるが、パソコンへインプリメントされる以前から、大型コンピュータやワークステーションで使われ、かなり長い歴史を持っている。従って、様々な言語分析に活用できるツールやプログラム類が豊富に流通しているのかと思ったが、それらは一般個人には開放されていないようで、上記文献などに見えるTutorial Sample以外は参照することができなかった。テキスト処理に威力を発揮すると言われ、言語計量やテキスト分析etc.に応用されてきたという経歴にもかかわらず、利用者自身が目的に応じて各種のプログラムを開発しなければならないのなら、既存資源の利用という面で、他の高級言語、Basic, Pascal, Cなどと較べて、とりわけ優位に立つとは言えないだろう。ただ、言語のシンタックスが柔軟で、他の汎用処理系では手続きの面倒なファイル入出力も手軽に扱える点などは見逃せない。文字列操作のための自己流の小さなツールに利用するには適しているだろう。反面、MS−DOS版SNOBOL4はインタプリタであるため、単独の実行ファイルで使えない他、やはり処理スピードの点で不満が残る。従って、プログラミングの容易さを最大限に活かしたSNOBOL4の利用環境は大型コンピュータなどの高速なハード上で実現できるのかもしれない。

スペイン語単語検索のためのSNOBLO4プログラムを（7）に示す。これをエディタ等でテキストファイル（ファイル名wsch.sno）として作成、保存した後、SONOBOL4インタプリタの本体プログラムと同じディレクトリでコマンドラインから

snobol4 wsch.sno

を入力すれば実行可能である。

wschはキーワードが含まれる文の総数を示した後、「文」データを表示するかどうかの確認を求める。Yまたはyを入力すると、頁・行数と共に各文全体を表示する。このプログラムでは発見データを1度配列に蓄えた後に、出力する形式をとっているので、扱い得るファイルの大きさ（wschでは発見文の上限を150としている）に限度があるが、テキスト走査中に検索単語を見つけ次第、逐次、文を表示するように変更すれば、プログラムはより短くなり、この制限を取り払うこともできる。

本稿で目指したような単語検索はテキスト処理用の常備ツールとして、いつでもすぐに使用できる状態にしておきたい性格のソフトである。ごく単純なこの種のコードでも、コンパイラにより実行型exeファイルにすると10−30Kbの大きさになる。SNOBOL, BASICなどの場合では、ソースプログラムは小さいがインタプリタ本体を併せると100Kb前後のスペースを占めてしまう。各種の常用ユーティリティがひしめくディスクの枚数を増さずに済ますためにはもっと軽量な方が望ましい。特に、ノート型パソコンの狭い不揮発ＲＡＭ上に置く場合は一層プログラム

(7)

```
*
*  wsch.sno  スペイン語ＳＴＸテキスト用の単語検索
*
*                    by Atsumi Deguchi          1990.8.09
*

        &ANCHOR = 1
        &TRIM = 1
        ORAC = ARRAY('0:150')
        PGLIN = ARRAY('0:150')
        UPPERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
        LOWERS = 'abcdefghijklmnopqrstuvwxyz'
        WRD = LOWERS UPPERS "'~^"
        DELIMIT = '.?:;!'
        WORDPAT = BREAK(WRD) SPAN(WRD) . WORD
        LNO = 0
        K = 0

        SCREEN = 'ファイル名を入力してください'
        INPUT('INP',1,,INPUT)              :F(END)
        SCREEN = '検索する単語を入力してください'
        PAT = INPUT                        :F(END)
        OLD = ''

READ    LINE = INP                         :F(HYOJI)
        LNO = LNO + 1
        LINE '[' RTAB(1) . PAGE :F(BUN)
        LNO = 0                            :(READ)
BUN     LINE BREAK(DELIMIT) . SENT LEN(1) . MARK REM . LAST :S(NEXT)
        &ANCHOR = 0
        LINE '-' RPOS(0) =                 :F(CONEC)
        &ANCHOR = 1
        OLD = OLD LINE                     :(READ)
CONEC   &ANCHOR = 1
        OLD  =  OLD  LINE ' '              :(READ)

NEXT    FRASE =  OLD  SENT
AGAIN   FRASE  WORDPAT =                   :F(ON)
        SWORD = REPLACE(WORD,UPPERS,LOWERS)
        IDENT(SWORD,PAT)                    :F(AGAIN)
        PGLIN<K> =  PAGE '-' LNO
        ORAC<K> = OLD  SENT MARK
        K = LE(K,149) K + 1                :F(IRR)

ON      LINE = LAST
        OLD = ''                           :(BUN)

IRR     SCREEN = '発見文が１５０を超えたので中断します'
HYOJI   SCREEN = EQ(K,0) '検索単語は発見されません'     :S(END)
        SCREEN = K '  文が発見されました'
        OUTPUT =
        SCREEN = '結果を出力しますか ? (Y/N)'
        INPUT  ('Y' ¦ 'y' )                :F(END)
        J = 0

NEXT2   OUTPUT = 'Page:Line  ' PGLIN<J>                 :F(END)
        OUTPUT = ORAC<J>                   :F(END)
        OUTPUT =
        J = LT(J,K - 1) J + 1              :S(NEXT2)
END
```

サイズが気にかかる。

そこで、検索スピードの向上と実行ファイルサイズの減少を図るため、アセンブリ言語によるスペイン語単語検索プログラムを試作した。（８Ｂ）のwfind.asmはそのソースコードで、これと（８Ａ）のマクロ定義ファイルsysmac.hをエディタ等で別々に作成した後、Turbo Assembler（ver 1.0以上）でアセンブルする。tasm wfind.asmでwfind.objを作り、次にリンカを起動して、tlink / t wfindで実行用プログラムwfind.comが生成される。装飾的要素を一切省いてあるだけでなく、入力に対するエラー指示なども切り詰めてあり、またガイダンス（使用説明）やヘルプ機能も付けてない。例えば、検索語が発見されないときは、そのまま終了して、特にメッセージを出さないし、またＨＩＴ語数も計算しない。しかし、前述の基本機能（６）は満たしながら高速性を確保したいという当初の目的は曲がりなりにも達成出来たと思う(3)。出来上がった.comファイルの大きさは2882byteであるが、実行ファイルを更にコンパクトにしたい場合は、入力バッファを小さくし、.exeモデルに変更してアセンブルし、最近話題の、実行ファイル圧縮プログラムlzexeによって１kbyte以下に軽量化することも可能である。wfindを起動するにはMS−DOSのプロンプトが出ている状態で

wfind　検索ファイル名

を投入するだけでよい。“Input search word”と表示されるので、検索語を入力してリターンを押し下げると、検索語が発見される毎にその所在ページ・行と「文」が表示されていく。なお、単語の全体一致のほか、頭部の部分一致も出来れば便利なので、＊によるゼロまたは任意数の文字にマッチする後方ワイルドカードのみ使用できるようにした。行数の計算はアルゴリズムのスピード化のために、文末を検出した時点を基準にしており、実際の語の出現する行とは一致しないことがある。２語以上の連語を検索する便を特に考慮していないが、もし「原データ」の語間スペースがすべて１個であり、行頭・行末に余分な空白がないならば、２行、２ページにわたる文内連語も検出可能である。

５．おわりに

スペイン語テキストファイルから単語を検索する基本作業に関わるいくつかの問題点を実用的な観点から取り上げてみた。不満な現状を多少とも軽減するため、汎用性があると思われる２つの短いソフトを自作した。今後、各種の処理のために様々なツールが作成公開され、少しでも便利な環境を共有するための協力体制が生まれることを期待したい。

（8A）

```
;  ------------------------------------------------------------
;
;       "sysmac.h"
;       wfind.asm のための
;       MS-DOS システムコール用　マクロ          1990.08.19
;
;  ------------------------------------------------------------

PUTCH   macro char
            mov ah,02h
            mov dl,char
            int 21h
            endm

PRINT   macro  mess
                mov ah,09h
                mov dx,offset mess
                int 21h
                endm

GETSTR  macro size,buff
                mov ah,0ah
                mov dx,offset buff
                mov buff,size
                int 21h
                endm

READH   macro hndl,buff,byte
                mov     bx,hndl
                mov     dx,offset buff
                mov     cx,byte
                mov     ah,3fh
                int     21h
                endm

OPENH   macro path,acc
                mov     dx,offset path
                mov     al,acc
                mov     ah,3dh
                int     21h
                endm

CLOSEH  macro hndl
                mov     bx,hndl
                mov     ah,3eh
                int     21h
                endm

RETDOS  macro
                mov ah,4ch
                int 21h
                endm
```

(8B)

```
;-----------------------------------------------------------------
;
;       wfind.asm スペイン語単語検索プログラム
;             (  .com ファイル版 )
;               ver 1.0     by Atsumi Deguchi          1990.08.19
;         NEC 9800 series MS-DOS       use Turbo Assembler ver 2.0
;
;                対象データ  :  ＳＴＸ形式平文テキスト
;
;-----------------------------------------------------------------
;
CR      equ     0dh
LF      equ     0ah
;
include sysmac.h
;
code    segment
        assume  cs:code,ds:code,es:code,ss:code
        org     80h

fname   db      128 dup(?)

Start:  mov     bl,fname
        xor     bh,bh
        mov     byte ptr fname[bx+1],0
        OPENH   fname+2,0
        jnc     Op
        PRINT   ermsg
        jmp     Fin
Op:     mov     handle,ax
        mov     si,offset temp
        mov     di,offset sent
        PRINT   msg
        GETSTR  40,buf
        PUTCH   LF
        xor     bx,bx
        mov     bl,[buf+1]
        mov     keyw[bx],'$'
Fr:     READH   handle,temp,1024
        jc      Fin
        cmp     ax,0
        je      Fin
        mov     bx,ax                   ;実際に読み込んだ字数
        mov     temp[bx],'$'
        mov     cx,ax
        cld
Lpm:    lodsb
        call    header
        cmp     al,LF
        je      Lpe
        cmp     al,CR
        jne     Skipcr
        inc     line
        mov     bh,[si-2]
        cmp     bh,'-'
        jne     Mrk
        dec     di
        jmp     Lpe
Mrk:    mov     al,' '
Skipcr: stosb
        cmp     al,'.'              ;文区切りの検査
        je      Disp
        cmp     al,':'
```

```
        je      Disp
        cmp     al,'?'
        je      Disp
        cmp     al,'!'
        je      Disp
        jmp     Lpe
;
Fin:    CLOSEH handle
        RETDOS
;
Disp:   mov     bl,'$'
        mov     [di],bl
        call    wmatch
        cmp     ax,1
        jne     Lp
        PRINT   hedpg
        mov     ax,line
        mov     bx,offset displfn-1
        push    si
        call    numtochar
        mov     bx,offset displin
        PRINT   displin
        pop     si
        PRINT   sent
        PUTCH CR
        PUTCH LF
Lp:     mov     di,offset sent
Lpe:    loop    Lpm
        mov     si,offset temp
        jmp     Fr
;
wmatch PROC                         ;単語パタンマッチ
        push    di
        push    si
        mov     si,offset sent
        mov     di,offset keyw
Schlst:
        mov     al,[si]
        cmp     al,[di]
        je      Next
        add     al,32               ;1字目は大文字も調べる
        cmp     al,[di]
        jne     Next2
Next:   jmp     Pattn
Next2:  inc     si
        mov     al,[si]
        cmp     al,'$'
        jne     Schlst
        mov     ax,0
        pop     si
        pop     di
        ret
Pattn:  mov     bl,[si-1]           ;1字前の字を保存
        inc     si
        inc     di
Lp1:    mov     al,[di]
        cmp     al,'$'              ;1字語の末尾をチェック
        je      Chk
        cmp     al,'*'              ;ワイルドカード
        je      Whed
        cmp     al,[si]
        jne     Skip
        inc     si
```

```
        inc     di
        cmp     al,'$'
        jne     Lp1
Chk:    mov     al,[si]
        call    isword
        je      Skip
Whed:   mov     al,bl
        call    isword
        jne     Hit
Skip:   mov     di,offset keyw
        jmp     Next2
Hit:    mov     ax,1
        pop     si
        pop     di
        ret
wmatch ENDP
;
isword  PROC                    ;語内文字か否か
        cmp     al,'z'
        jbe     Smll
        cmp     al,'~'
        jz      Fn
Smll:   cmp     al,'a'
        jae     Yes
        cmp     al,'^'
        jz      Fn
        cmp     al,'Z'
        ja      Fn
        cmp     al,'A'
        jae     Yes
        cmp     al,27h
        jmp     Fn
Yes:    cmp     al,al
Fn:     ret
isword   ENDP
;
numtochar PROC                  ;数値を10進文字列に変換
        mov     si,10
Cntlp:
        sub     dx,dx
        div     si
        add     dl,'0'
        mov     [bx],dl
        dec     bx
        cmp     ax,0
        jne     Cntlp
        mov     dl,' '
        mov     [bx],dl         ;左に空白を入れる
        dec     bx              ;左桁あふれ注意！
        mov     [bx],dl
        ret
numtochar ENDP
;
header PROC                     ;頁行の参照部処理
        cmp     al,'['
        jne     Hde
        lodsb
        dec     cx
        xor     bx,bx
Hrep:   mov     hedlab[bx],al
        inc     bx
        lodsb
        dec     cx
```

```
        cmp     al,']'
        jne     Hrep
        lodsb
        dec     cx
        mov     hedlab[bx],'$'
        mov     line,0
        jmp     Lpe
Hde:    ret
header ENDP
;
msg     db      'Input Search Word: ','$'
buf     db      2 dup(?)
keyw    db      40 dup(?)
line    dw      1
temp    db      1025 dup(?)
dmmy    db      ' '                 ;文頭語を認識する偽区切り
sent    db      1281 dup(?)
handle  dw      ?
displin db      '  Line : ', 5 dup(' ')
displfn db      CR,LF,'$'
hedpg   db      'Page : '
hedlab  db      40 dup(' ')
ermsg   db      "Can't open data file",'$'
;
code    ends
        end     Start
```

注

1．本稿で使ったGREPはBorland社のプログラミング言語に付属するユーティリティに含まれたgrep.com 7029 bytes 90-07-09 2:00である。

2．Vanilla SNOBOL4 ver. 2.14で、Catspaw社のSNOBOL4＋のentry-level versionである。PC－DOS用であるが、ＮＥＣのMS－DOS ver. 3.1で問題なく動作した。

3．同じハード条件下でＲＡＭディスク上にある約30kbの平文テキストで、hit数１文の検索例の所要時間を計ったところ、wschで約50.1秒、wfindで約0.9秒であった。後者の方が50倍程高速である。ただし、hit数が増えれば、画面（ファイル）出力処理の方がコスト要因になるので両者の速度倍率の差はずっと縮まる。

（1990年８月15日）

REFERENCES

Butler, Christopher (1985). Computers in Linguistics. Basil Blackwell, Oxford.

Griswold, R.E., J.F.Poage & I.P.Polonsky (1970). The SNOBOL4 Programming Language. Prentice-Hall, Englewood Cliff.

Hockey, Susan (1985). Snobol Programming for the Humanities. Clarendon Press Oxford.［邦訳 “SNOBOL入門 テキスト処理のためのプログラミング” 昭和63年、丸善(株)］

出口厚実（1989）「スペイン語テキスト・データとパーソナルコンピュータの使用環境」—Estudios Hispánicos 14, pp.1－13

————（1990）「スペイン語テキスト・ファイルの作成とファイル形式の変換」—Estudios Hispánicos 15, pp.1－15

（1990. 9. 5 受理）